COMPLAINT EXHIBIT 13
U.S. Patent No. 8,050,321 (AV1)

As demonstrated in the chart below, ASUS directly and indirectly infringes at least claim 8 U.S. Patent No. 8,050,321 (the "'321 Patent"). ASUS directly infringes, contributes to the infringement of, and/or induces infringement of the '321 Patent by making, using, selling, offering for sale, and/or importing into the United States the Accused Products that are covered by one or more claims of the '321 Patent. The Accused Products are devices that decode AV1-compliant video. For example, the ASUS Q543MV Notebook ("ASUS Q543MV") is a representative product for other ASUS devices that use graphics processing units that decode AV1-compliant video.

The ASUS Q543MV contains at least one video decoder that helps decode AV1-compliant video.[1] While evidence from the ASUS Q543MV is specifically charted herein, the evidence and contentions charted herein apply equally to the other ASUS Accused Products that decode AV1-compliant video.

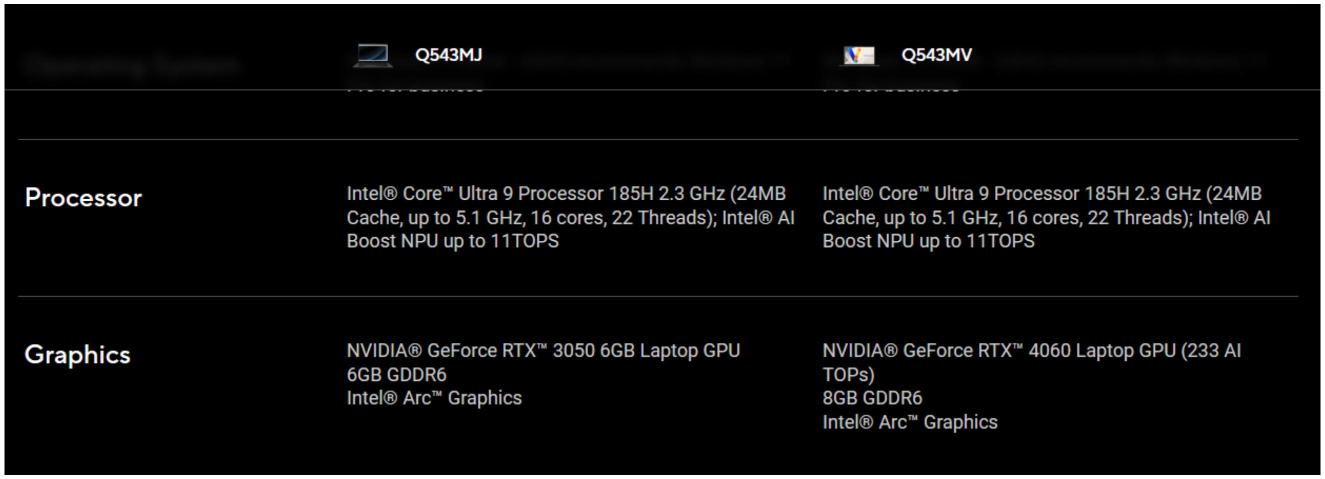
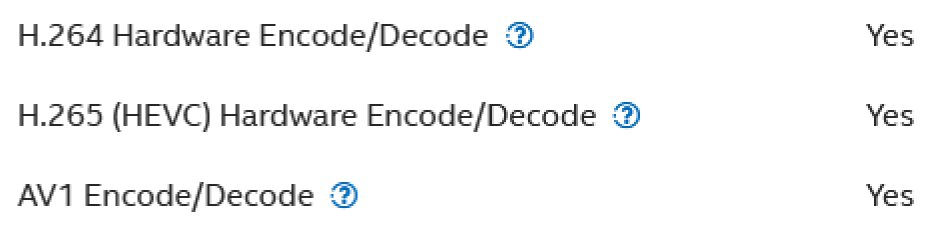
No part of this exemplary chart construes, or is intended to construe, the specification, file history, or claims of the '321 Patent. Moreover, this exemplary chart does not limit, and is not intended to limit, Nokia's infringement positions or contentions.

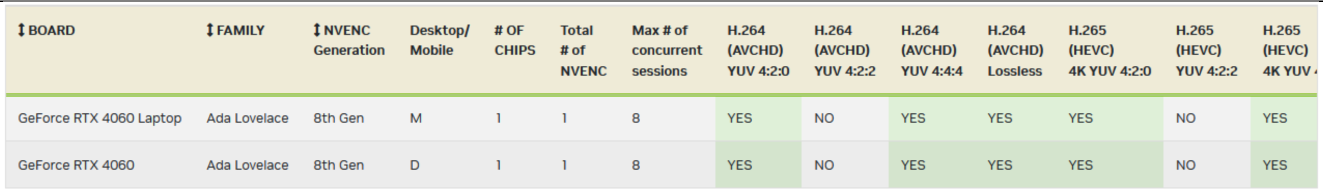
The following infringement chart includes exemplary citations to AV1 Bitstream & Decoding Process Specification (available at https://aomediacodec.github.io/av1-spec/av1-spec.pdf) (the "AV1 Specification"). Any ASUS device that includes a decoder that practices the functionality in any of these editions of the AV1 Specification practices the claims of the '321 Patent. As shown in the chart below, each of the Accused Products includes at least one decoder for decoding encoded AV1-compliant video ("AV1 Decoder") to practice the AV1 Specification and is covered by the claims of '321 Patent.

Nokia contends each of the following limitations is met literally, and, to the extent a limitation is not met literally, it is met under the doctrine of equivalents.[2]

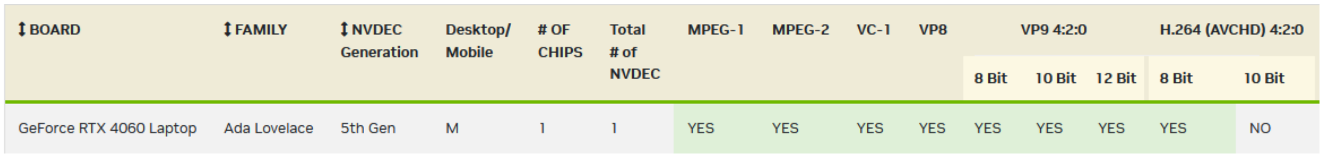
[1] See, e.g., https://www.asus.com/us/laptops/for-home/everyday-use/asus-vivobook-pro-15-oled-q543/techspec/; https://www.intel.com/content/www/us/en/products/sku/236849/intel-core-ultra-9-processor-185h-24m-cache-up-to-5-10-ghz/specifications.html; https://developer.nvidia.com/video-encode-and-decode-gpu-support-matrix-new.

[2] This claim chart is based on the information currently available to Nokia and is intended to be exemplary in nature. Nokia reserves all rights to update and elaborate its infringement positions, including as Nokia obtains additional information during discovery.

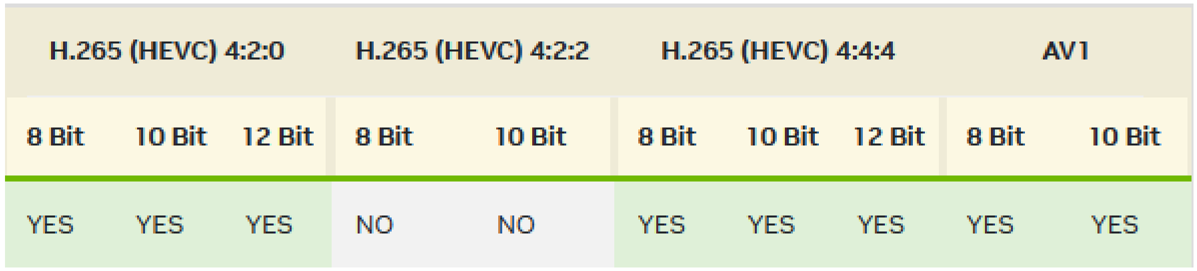
U.S. Patent No. 8,050,321	ASUS Accused Products
8. [A] A method for decoding a compressed video sequence,	Each of the Accused Products, such as the ASUS Q543MV, performs a method for decoding a compressed video sequence. For example, and without limitation, the Asus Q543MV uses hardware-accelerated decoding and includes an NVIDIA GeForce RTX 4060 Laptop graphics processing unit ("GPU") and an Intel Core Ultra 9 Processor 185H.  | | Q543MJ | Q543MV | | Processor | Intel® Core™ Ultra 9 Processor 185H 2.3 GHz (24MB Cache, up to 5.1 GHz, 16 cores, 22 Threads); Intel® AI Boost NPU up to 11TOPS | Intel® Core™ Ultra 9 Processor 185H 2.3 GHz (24MB Cache, up to 5.1 GHz, 16 cores, 22 Threads); Intel® AI Boost NPU up to 11TOPS | | Graphics | NVIDIA® GeForce RTX™ 3050 6GB Laptop GPU 6GB GDDR6 Intel® Arc™ Graphics | NVIDIA® GeForce RTX™ 4060 Laptop GPU (233 AI TOPs) 8GB GDDR6 Intel® Arc™ Graphics | Source: https://www.asus.com/us/laptops/for-home/everyday-use/asus-vivobook-pro-15-oled-q543/techspec/ (last accessed March 6, 2025).  H.264 Hardware Encode/Decode — Yes H.265 (HEVC) Hardware Encode/Decode — Yes AV1 Encode/Decode — Yes Source: https://www.intel.com/content/www/us/en/products/sku/236849/intel-core-ultra-9-processor-185h-24m-cache-up-to-5-10-ghz/specifications.html (last accessed March 6, 2025)(specifications for Intel Core Ultra 9 185H).



↕BOARD	↕FAMILY	↕NVENC Generation	Desktop/ Mobile	# OF CHIPS	Total # of NVENC	Max # of concurrent sessions	H.264 (AVCHD) YUV 4:2:0	H.264 (AVCHD) YUV 4:2:2	H.264 (AVCHD) YUV 4:4:4	H.264 (AVCHD) Lossless	H.265 (HEVC) 4K YUV 4:2:0	H.265 (HEVC) YUV 4:2:2	H.265 (HEVC) 4K YUV
GeForce RTX 4060 Laptop	Ada Lovelace	8th Gen	M	1	1	8	YES	NO	YES	YES	YES	NO	YES
GeForce RTX 4060	Ada Lovelace	8th Gen	D	1	1	8	YES	NO	YES	YES	YES	NO	YES



↕BOARD	↕FAMILY	↕NVDEC Generation	Desktop/ Mobile	# OF CHIPS	Total # of NVDEC	MPEG-1	MPEG-2	VC-1	VP8	VP9 4:2:0			H.264 (AVCHD) 4:2:0	
										8 Bit	10 Bit	12 Bit	8 Bit	10 Bit
GeForce RTX 4060 Laptop	Ada Lovelace	5th Gen	M	1	1	YES	YES	YES	YES	YES	YES	YES	YES	NO



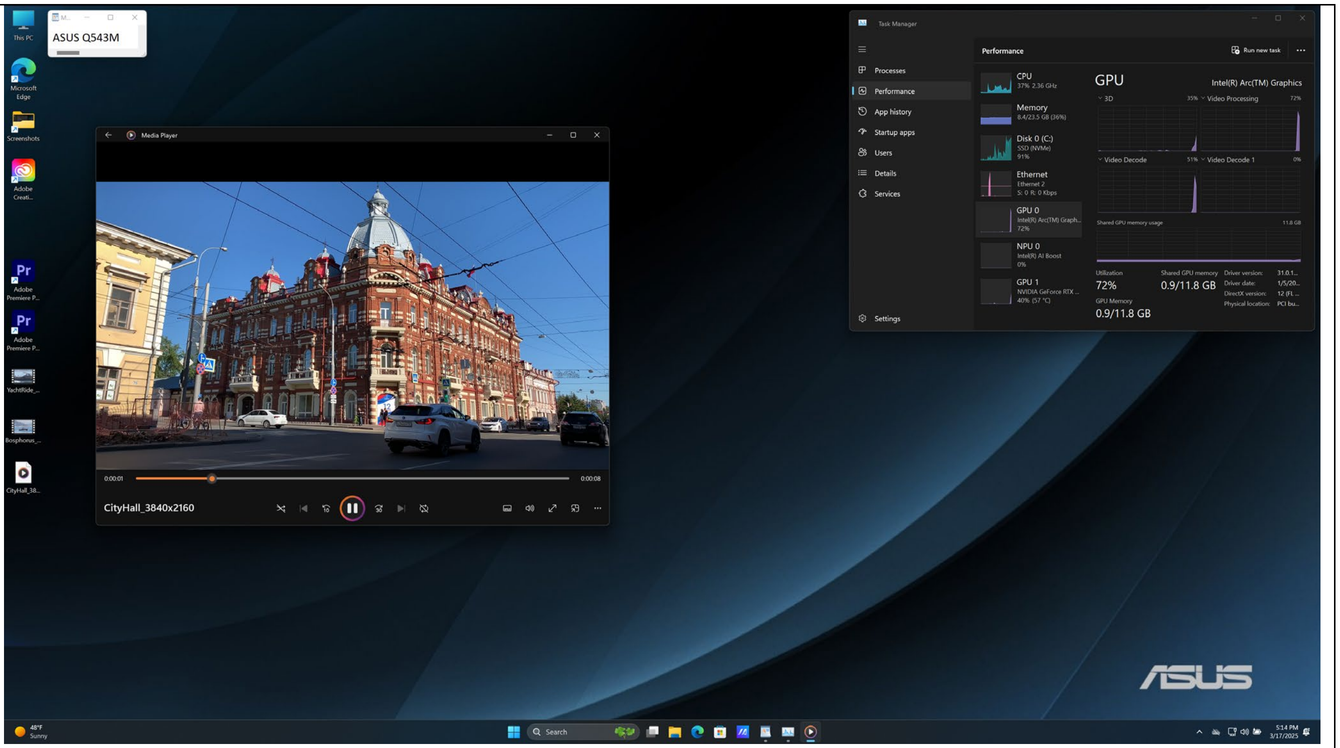
H.265 (HEVC) 4:2:0			H.265 (HEVC) 4:2:2		H.265 (HEVC) 4:4:4			AV1	
8 Bit	10 Bit	12 Bit	8 Bit	10 Bit	8 Bit	10 Bit	12 Bit	8 Bit	10 Bit
YES	YES	YES	NO	NO	YES	YES	YES	YES	YES

Source: https://developer.nvidia.com/video-encode-and-decode-gpu-support-matrix-new (last accessed March 6, 2025)(row for 4060 Laptop GPU).

The ASUS Q543MV, includes at least one AV1 Decoder for performing a method of decoding a compressed video sequence.

For example and without limitation, the AV1 Specification specifies a decoding process that reads a bitstream representing coded pictures of a video sequence and derives decoded pictures from it.

For example, an ASUS Q543MV was used to play back an AV1-compliant video.

	 ASUS Q543M Media Player CityHall_3840x2160 Task Manager Performance GPU Intel(R) Arc(TM) Graphics Video Decode 72% 0.9/11.8 GB Source: Screenshot of AV1-compliant video playback on ASUS Q543MV.
[B] the method comprising: decoding from the video sequence an indication of at least one image frame, which is the first image frame, in decoding order, of an independent sequence, wherein all motion-compensated temporal prediction references of the independent sequence refer only to image frames	Each of the Accused Products, such as the ASUS Q543MV, performs a method of decoding a compressed video sequence, the method comprising: decoding from the video sequence an indication of at least one image frame, which is the first image frame, in decoding order, of an independent sequence, wherein all motion-compensated temporal prediction references of the independent sequence refer only to image frames within said independent sequence. For example and without limitation, the AV1 Specification also specifies decoding from the video sequence an indication of at least one image frame, which is the first image frame, in decoding order, of an independent sequence, wherein all motion-compensated temporal prediction references of the independent sequence refer only to image frames within said independent sequence. For example, in the AV1 Specification, key frames begin an independent sequence. Key frames are intra frames that "reset" the decoding process. Sequences starting with a key frame are independent of frames

within said independent sequence;	that precede the key frame and do not rely on those earlier frames for reference. This provides, e.g., error resiliency. **7.5. Ordering of OBUs** A bitstream conforming to this specification consists of one or more coded video sequences. A coded video sequence consists of one or more temporal units. A temporal unit consists of a series of OBUs starting from a temporal delimiter, optional sequence headers, optional metadata OBUs, a sequence of one or more frame headers, each followed by zero or more tile group OBUs as well as optional padding OBUs. A new coded video sequence is defined to start at each temporal unit which satisfies both of the following conditions: • A sequence header OBU appears before the first frame header. • The first frame header has frame_type equal to KEY_FRAME, show_frame equal to 1, show_existing_frame equal to 0, and temporal_id equal to 0. (AV1 Bitstream & Decoding Process Specification, at 204). **2. Terms and definitions** **Key frame** An Intra frame which resets the decoding process when it is shown. (AV1 Bitstream & Decoding Process Specification, at 3). **6.8.14. Tile info semantics** Each of these data values for each segment may be individually updated at the frame level. Where a value is not updated in a given frame, the value from the previous frame persists. The exceptions to this are key frames, intra only frames or other frames where independence from

past frame values is required (for example to enable error resilience). In such cases all values are reset as described in the semantics for setup_past_independence.

(AV1 Bitstream & Decoding Process Specification, at 164).

7.5. Ordering of OBUs

Otherwise (show_existing_frame is equal to 1), if frame_type is equal to KEY_FRAME, the reference frame loading process as specified in section 7.21 is invoked (this process loads frame state from the reference frames into the current frame state variables).

The following ordered steps now apply:

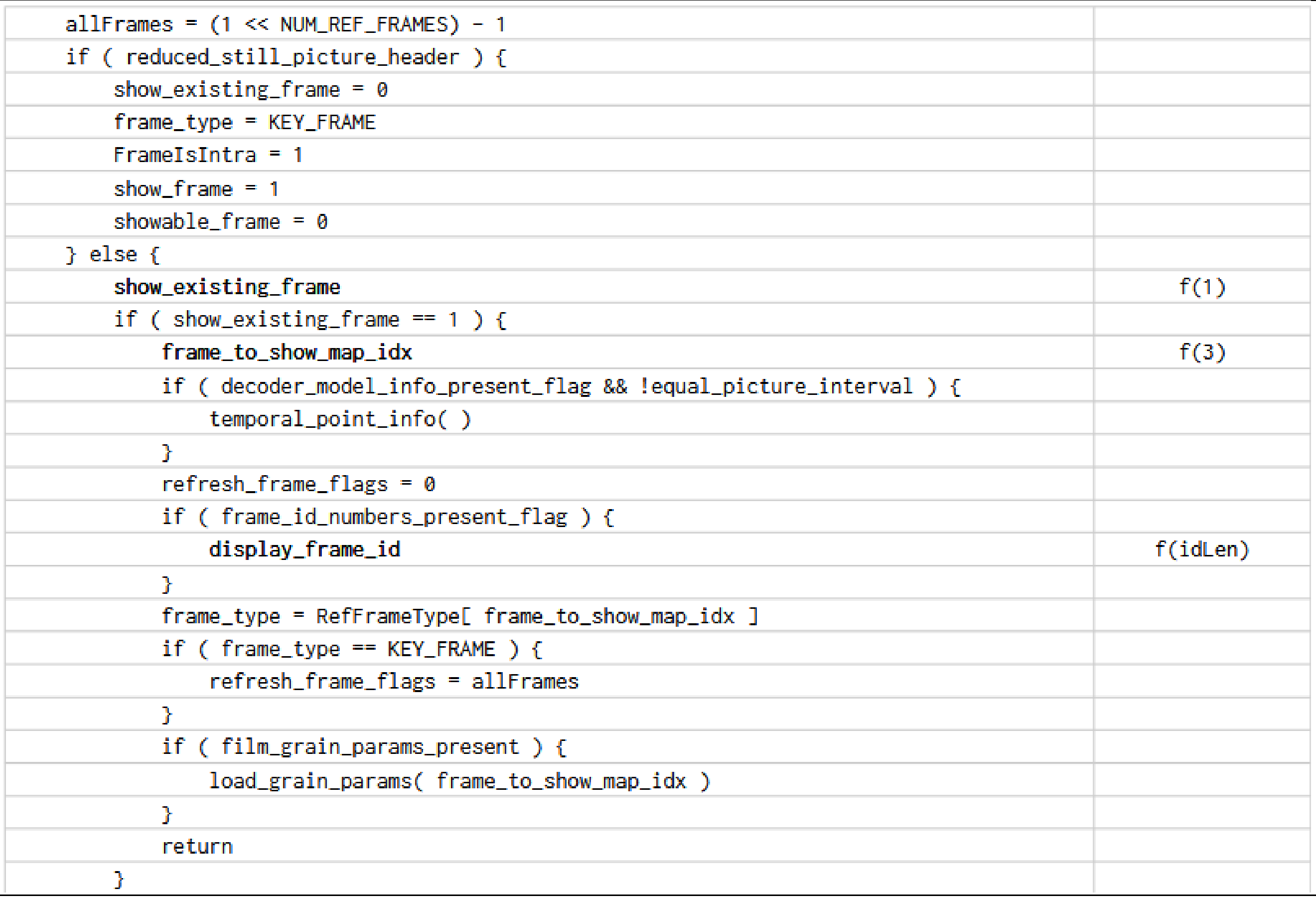
1. The reference frame update process as specified in section 7.20 is invoked (this process saves the current frame state into the reference frames).

2. If show_frame is equal to 1 or show_existing_frame is equal to 1, the output process as specified in section 7.18 is invoked (this will output the current frame or a saved frame).

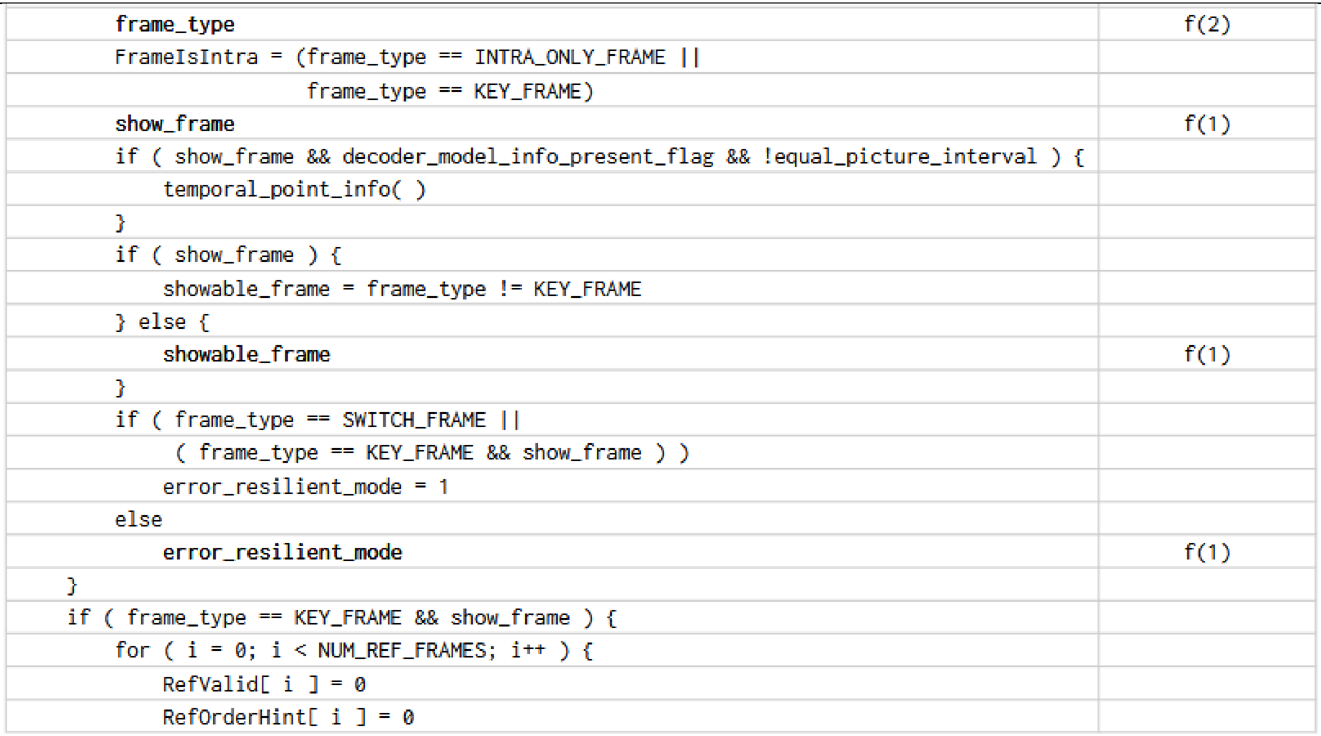
(AV1 Bitstream & Decoding Process Specification, at 204).

5.9.2. Uncompressed header syntax

uncompressed_header( ) {	Type
if (frame_id_numbers_present_flag) {	
idLen = (additional_frame_id_length_minus_1 +	
delta_frame_id_length_minus_2 + 3)	
}	



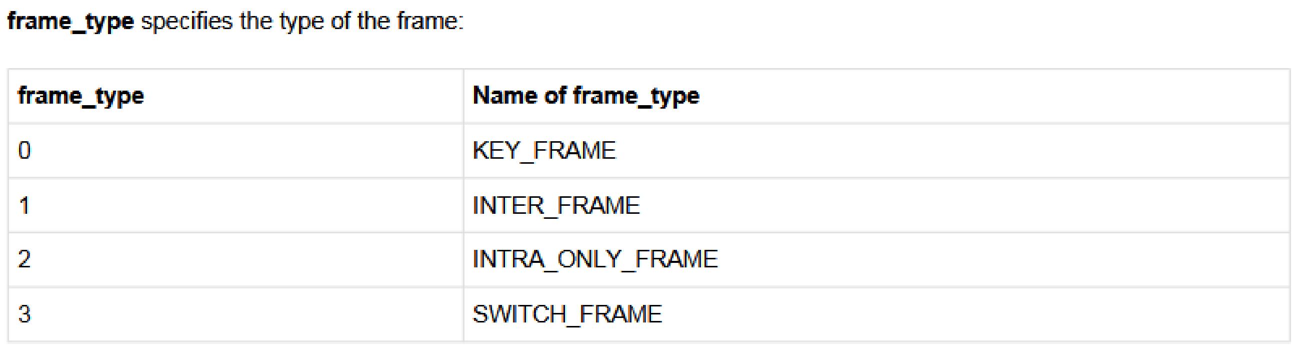
```
  allFrames = (1 << NUM_REF_FRAMES) - 1
  if ( reduced_still_picture_header ) {
      show_existing_frame = 0
      frame_type = KEY_FRAME
      FrameIsIntra = 1
      show_frame = 1
      showable_frame = 0
  } else {
      show_existing_frame                                                          f(1)
      if ( show_existing_frame == 1 ) {
          frame_to_show_map_idx                                                    f(3)
          if ( decoder_model_info_present_flag && !equal_picture_interval ) {
              temporal_point_info( )
          }
          refresh_frame_flags = 0
          if ( frame_id_numbers_present_flag ) {
              display_frame_id                                                     f(idLen)
          }
          frame_type = RefFrameType[ frame_to_show_map_idx ]
          if ( frame_type == KEY_FRAME ) {
              refresh_frame_flags = allFrames
          }
          if ( film_grain_params_present ) {
              load_grain_params( frame_to_show_map_idx )
          }
          return
      }
```



```
        frame_type                                                                      f(2)
        FrameIsIntra = (frame_type == INTRA_ONLY_FRAME ||
                        frame_type == KEY_FRAME)
        show_frame                                                                      f(1)
        if ( show_frame && decoder_model_info_present_flag && !equal_picture_interval ) {
            temporal_point_info( )
        }
        if ( show_frame ) {
            showable_frame = frame_type != KEY_FRAME
        } else {
            showable_frame                                                              f(1)
        }
        if ( frame_type == SWITCH_FRAME ||
             ( frame_type == KEY_FRAME && show_frame ) )
            error_resilient_mode = 1
        else
            error_resilient_mode                                                        f(1)
    }
    if ( frame_type == KEY_FRAME && show_frame ) {
        for ( i = 0; i < NUM_REF_FRAMES; i++ ) {
            RefValid[ i ] = 0
            RefOrderHint[ i ] = 0
```

(AV1 Bitstream & Decoding Process Specification, at 38).

6.8.2. Uncompressed header semantics



frame_type specifies the type of the frame:

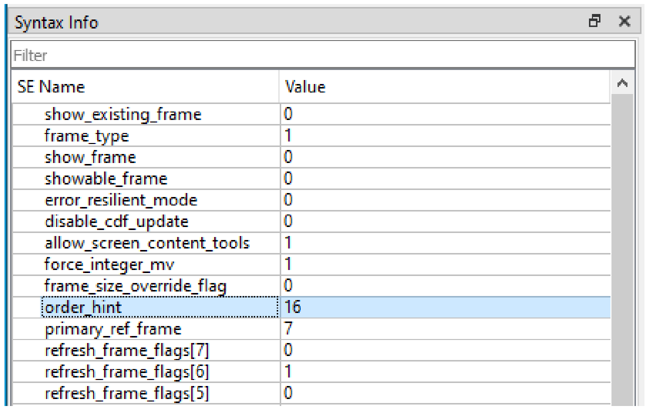
frame_type	**Name of frame_type**
0	KEY_FRAME
1	INTER_FRAME
2	INTRA_ONLY_FRAME
3	SWITCH_FRAME

(AV1 Bitstream & Decoding Process Specification, at 150).

	Further, the evidence cited for claim limitation 8[A] applies to this claim limitation.
[C] starting the decoding of the video sequence from said first image frame of the independent sequence, whereby the video sequence is decoded without prediction from any image frame decoded prior to said first image frame;	Each of the Accused Products, such as the ASUS Q543MV, performs a method of decoding a compressed video sequence, the method comprising: starting the decoding of the video sequence from said first image frame of the independent sequence, whereby the video sequence is decoded without prediction from any image frame decoded prior to said first image frame. For example and without limitation, the AV1 Specification also specifies starting the decoding of the video sequence from said first image frame of the independent sequence, whereby the video sequence is decoded without prediction from any image frame decoded prior to said first image frame. For example, the AV1 Decoder starts the decoding of the video sequence from a key frame (the first image frame of the independent sequence). The video sequence is then decoded without prediction from any frames prior to the key frame. **2. Terms and definitions** **Key frame** An Intra frame which resets the decoding process when it is shown. (AV1 Bitstream & Decoding Process Specification, at 3). **6.8.13. Segmentation params semantics** Each of these data values for each segment may be individually updated at the frame level. Where a value is not updated in a given frame, the value from the previous frame persists. The exceptions to this are key frames, intra only frames or other frames where independence from past frame values is required (for example to enable error resilience). In such cases all values are reset as described in the semantics for setup_past_independence. (AV1 Bitstream & Decoding Process Specification, at 164).

	7.6.3. Conformance requirements Informally, the requirement for decoder conformance is that decoding can start at any key frame random access point or delayed random access point. The rest of this section makes this requirement more precise. Starting at a key frame random access point is trivial, because if the earlier temporal units are dropped, the remaining temporal units still constitute a valid bitstream. Starting at a delayed random access point is harder to define because: • if all temporal units before the key frame dependent recovery point are dropped, it is impossible to decode (because the relevant delayed random access point has been dropped) • if all temporal units before the delayed random access point are dropped, it is unclear what should happen for frames between the delayed random access point and the key-frame dependent recovery point (some applications may wish these to be dropped, while others may wish them to be displayed) • in either case, the remaining temporal units do not constitute a valid standalone bitstream (because it does not start with a shown key frame) (AV1 Bitstream & Decoding Process Specification, at 207). Further, the evidence cited for claim limitations 8[A-B] applies to this claim limitation.
[D] decoding identifier values for image frames according to a numbering scheme; and	Each of the Accused Products, such as the ASUS Q543MV, performs a method of decoding a compressed video sequence, the method comprising: decoding identifier values for image frames according to a numbering scheme. For example and without limitation, the AV1 Specification also specifies decoding identifier values for image frames according to a numbering scheme.

For example, the AV1 Decoder decodes identifier values for image frames according to a number scheme. For example, the order_hint field provides identifier values for image frames according to a numbering scheme. An example of order_hint is shown below.



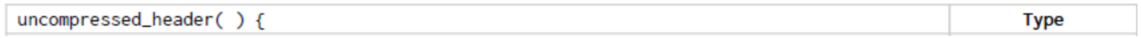
Syntax Info

SE Name	Value
show_existing_frame	0
frame_type	1
show_frame	0
showable_frame	0
error_resilient_mode	0
disable_cdf_update	0
allow_screen_content_tools	1
force_integer_mv	1
frame_size_override_flag	0
order_hint	16
primary_ref_frame	7
refresh_frame_flags[7]	0
refresh_frame_flags[6]	1
refresh_frame_flags[5]	0

Source: VQ Analyzer testing of AV1 sample reference video.

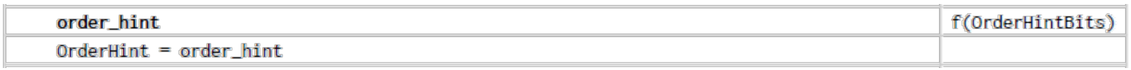
The AV1 Specification further describes how an AV1 decoder such as the accused products decodes identifier values for image frames according to a numbering scheme:

5.9.2. Uncompressed header syntax



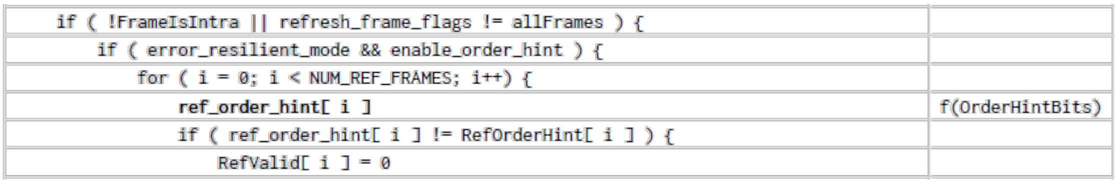
uncompressed_header( ) {	Type

. . .



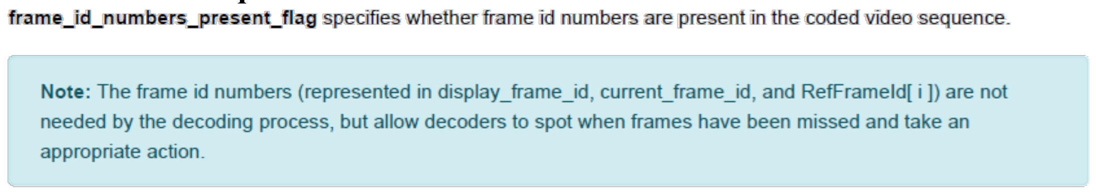
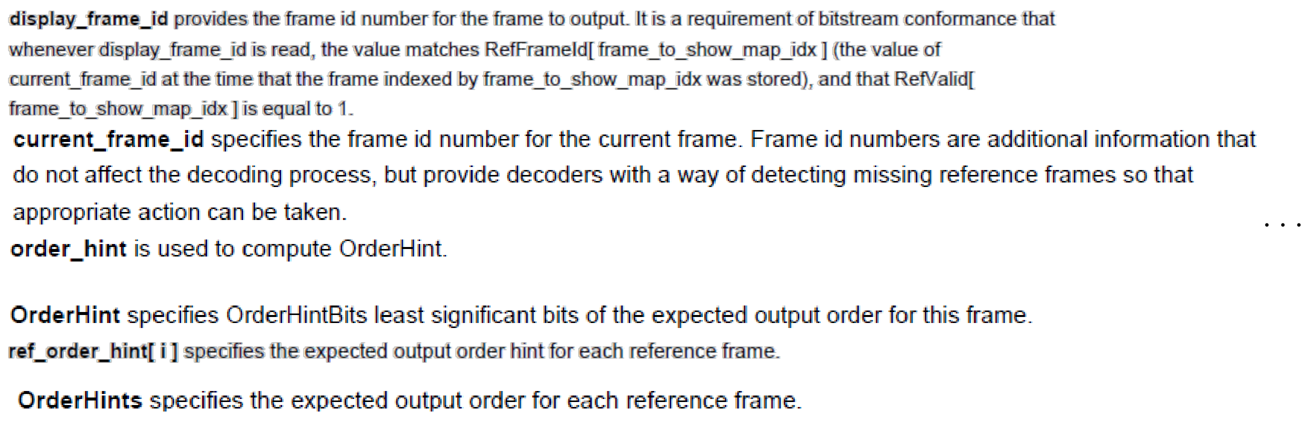
order_hint	f(OrderHintBits)
OrderHint = order_hint	

. . .



if (!FrameIsIntra || refresh_frame_flags != allFrames) {	
if (error_resilient_mode && enable_order_hint) {	
for (i = 0; i < NUM_REF_FRAMES; i++) {	
ref_order_hint[i]	f(OrderHintBits)
if (ref_order_hint[i] != RefOrderHint[i]) {	
RefValid[i] = 0	

(AV1 Bitstream & Decoding Process Specification, at 37-40)

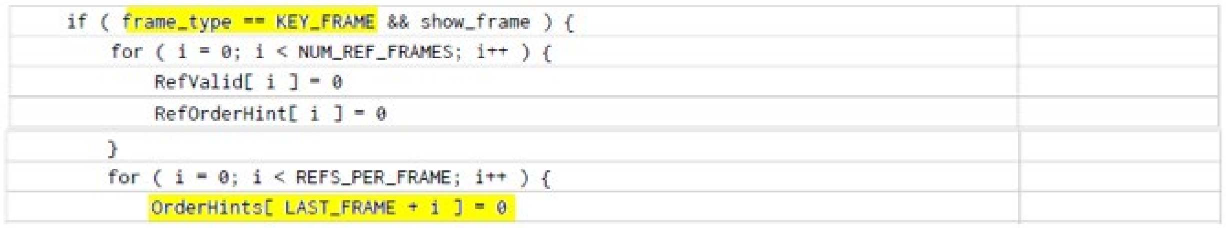
	6.4.1. General sequence header OBU semantics  **frame_id_numbers_present_flag** specifies whether frame id numbers are present in the coded video sequence. **Note:** The frame id numbers (represented in display_frame_id, current_frame_id, and RefFrameId[i]) are not needed by the decoding process, but allow decoders to spot when frames have been missed and take an appropriate action. (AV1 Bitstream & Decoding Process Specification, at, 115). **6.8.2. Uncompressed header semantics**  **display_frame_id** provides the frame id number for the frame to output. It is a requirement of bitstream conformance that whenever display_frame_id is read, the value matches RefFrameId[frame_to_show_map_idx] (the value of current_frame_id at the time that the frame indexed by frame_to_show_map_idx was stored), and that RefValid[frame_to_show_map_idx] is equal to 1. **current_frame_id** specifies the frame id number for the current frame. Frame id numbers are additional information that do not affect the decoding process, but provide decoders with a way of detecting missing reference frames so that appropriate action can be taken. . . . **order_hint** is used to compute OrderHint. **OrderHint** specifies OrderHintBits least significant bits of the expected output order for this frame. **ref_order_hint[i]** specifies the expected output order hint for each reference frame. **OrderHints** specifies the expected output order for each reference frame. (AV1 Bitstream & Decoding Process Specification, at, 150-54). Further, the evidence cited for claim limitations 8[A-C] applies to this claim limitation.
[E] resetting the identifier value for the indicated first image frame of the independent sequence.	Each of the Accused Products, such as the ASUS Q543MV, performs a method of decoding a compressed video sequence, the method comprising: resetting the identifier value for the indicated first image frame of the independent sequence. For example and without limitation, the AV1 Specification also specifies resetting the identifier value for the indicated first image frame of the independent sequence. **2. Terms and definitions**

Key frame

An Intra frame which resets the decoding process when it is shown.

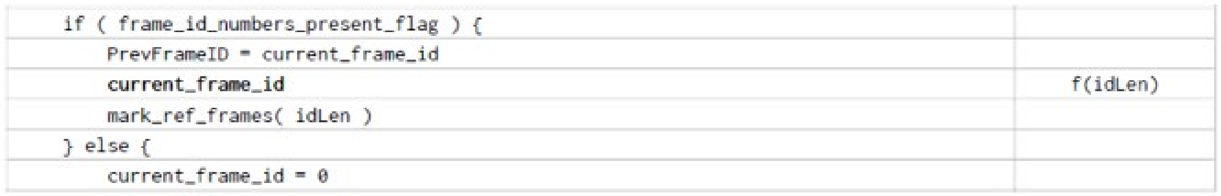
(AV1 Bitstream & Decoding Process Specification, at 3).

5.9.2. Uncompressed header syntax



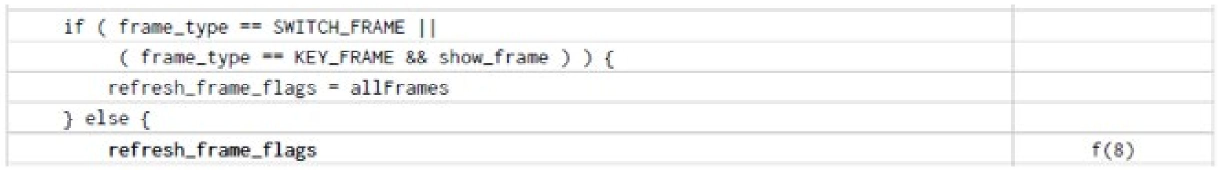
```
if ( frame_type == KEY_FRAME && show_frame ) {
    for ( i = 0; i < NUM_REF_FRAMES; i++ ) {
        RefValid[ i ] = 0
        RefOrderHint[ i ] = 0
    }
    for ( i = 0; i < REFS_PER_FRAME; i++ ) {
        OrderHints[ LAST_FRAME + i ] = 0
```

. . .



```
if ( frame_id_numbers_present_flag ) {
    PrevFrameID = current_frame_id
    current_frame_id                                    f(idLen)
    mark_ref_frames( idLen )
} else {
    current_frame_id = 0
```

. . .



```
if ( frame_type == SWITCH_FRAME ||
     ( frame_type == KEY_FRAME && show_frame ) ) {
    refresh_frame_flags = allFrames
} else {
    refresh_frame_flags                                 f(8)
```

(AV1 Bitstream & Decoding Process Specification, at, 38-40).

Further, the evidence cited for claim limitations 8[A-D] applies to this claim limitation.